Gene Expression and Fast Construction of Distributed Evolutionary Representation

Hillol Kargupta
hillol@eecs.wsu.edu
School of EECS, Washington State University, Pullman, WA 99164-2752, USA

Byung-Hoon Park
bhpark@eecs.wsu.edu
School of EECS, Washington State University, Pullman, WA 99164-2752, USA

Abstract
The gene expression process in nature produces different proteins in different cells from different portions of the DNA. Since proteins control almost every important activity in a living organism, at an abstract level, gene expression can be viewed as a process that evaluates the merit or “fitness” of the DNA. This distributed evaluation of the DNA would not be possible without a decomposed representation of the fitness function defined over the DNAs. This paper argues that, unless the living body was provided with such a representation, we have every reason to believe that it must have an efficient mechanism to construct this distributed representation. This paper demonstrates polynomial-time computability of such a representation by proposing a class of efficient algorithms. The main contribution of this paper is two-fold. On the algorithmic side, it offers a way to scale up evolutionary search by detecting the underlying structure of the search space. On the biological side, it proves that the distributed representation of the evolutionary fitness function in gene expression can be computed in polynomial-time. It advances our understanding about the representation construction in gene expression from the perspective of computing. It also presents experimental results supporting the theoretical performance of the proposed algorithms.

Keywords
Gene expression, representation construction, Walsh analysis, linkage learning.

1 Introduction
The last fifty years witnessed the gradual development of different adaptive search algorithms that use motivations from natural evolution. The genetic algorithms (GAs) (Holland, 1975), evolutionstrategie (ES) (Rechenberg, 1964), evolutionary programming (EP) (Fogel et al., 1966), and genetic programming (GP) (Koza, 1992) are some examples that found many successful applications in search, optimization, and machine learning. These algorithms are primarily motivated by the passage of evolutionary information from generation to generation through selection, crossover, and mutation operations. Evaluation of fitness in GAs, ES, EP, and GP is often a simple one-step process of computing the objective function (with some exceptions (Schackleton et al., 2000)). However, this is not quite that simple in nature. This process (often called the gene expression) can be arguably viewed as a process that significantly contributes towards evaluating the evolutionary fitness of a genome through a sequence of representation transformations (DNA→mRNA→Protein). This paper questions the simplistic and centralized one-step event of the objective function evaluation often adopted in the practice of evolutionary algorithms. It draws attention to the gene expression process and argues...
that this complex process may have a non-trivial role in controlling the efficiency of the evolutionary search.

Although today we have a reasonable idea about the main biological steps of gene expression, we know very little about its role in evolutionary search. This paper takes a modest step in unraveling this mystery. It explores the computability of the decomposed representation of the evolutionary fitness function exploited during gene expression in the light of our existing understanding about inductive construction of function structure. It is not a paper on GAs, GP, or any other popular paradigm of evolutionary algorithms. However, it may have important implication in the future of evolutionary computation.

Gene expression involves construction of proteins from the DNA sequence. Since proteins play an important role in almost every major activity of a living body, the efficacy of the produced proteins determines its overall fitness. So at a very abstract level, gene expression can be viewed as a process that evaluates the evolutionary fitness. Interestingly enough, it does so in a very distributed fashion. Different portions of the DNA, the evolutionary information carrier, are evaluated in different cells through the production of different proteins. This paper suggests that the distributed evaluation of evolutionary fitness may have deeper implications. Distributed evaluation is possible only when the underlying fitness function can be represented in a decomposed form. Such representations also expose the underlying decomposability of the fitness function, and therefore it is important for the scalability of any search algorithm, including evolutionary search (Kargupta and Goldberg, 1996). Therefore, one must wonder if the gene expression process has a hidden mechanism to construct such representations. However, we must address a more fundamental question before exploring that intriguing possibility. We should first show that such a construction is feasible using our basic understanding of computation. This paper shows that this is indeed possible in polynomial-time when the number of mutually interacting genes is bounded regardless of the length of the genome.

Section 2 discusses the role of function induction in learning, adaptation, and optimization. Section 3 discusses the distributed fitness evaluation in the natural gene expression process and argues the need for an efficient function induction mechanism in our model of evolutionary computation. Section 4 discusses the previous work in evolutionary computation that considered gene expression. Section 5 briefly overviews the basics of orthonormal basis representation. Section 6 presents two algorithms to compute a distributed representation using orthonormal basis in polynomial time. Section 7 discusses the application of the proposed algorithms in detecting the search space structure. Section 8 presents experimental results of applying this algorithm to detect the underlying structure of different, difficult optimization problems. Section 9 discusses the future and on-going work and Section 10 concludes the paper.

2 Function Induction in Learning, Adaptation, and Optimization

Function induction plays an important role in machine learning, adaptation, and non-enumerative black-box optimization. In function induction, the goal is to learn a function \( \hat{f} : X^n \rightarrow Y \) from the data set \( \Omega = \{(x_{(1)}, y_{(1)}), (x_{(2)}, y_{(2)}), \ldots, (x_{(k)}, y_{(k)})\} \) generated by some underlying function \( f : X^n \rightarrow Y \), such that \( \hat{f} \) approximates \( f \). Any member of the domain \( x = x_1, x_2, \ldots, x_\ell \) is an \( \ell \)-tuple and \( x_j \)s correspond to individual feature variables of the domain. Learning a function in a chosen form or representation is relevant to many important problem domains.

Empirical machine learning (Michalski, 1983) is directly related to function induc-
tion. Given a set of observed behaviors, the goal of empirical machine learning is to learn a function that closely approximates those behaviors, which is essentially the function induction problem itself when the behavior is viewed as a function defined over the domain of different situations.

Function induction also plays an important role in natural and artificial adaptation. The word adaptation literally means “to fit to” (ad + aptare). In general, we call a system adaptive when it can exhibit appropriate behavioral dynamics in response to changes in an uncertain environment. Inductive function learning has a close relation to adaptation (Holland, 1975). Adaptation in uncertain environments typically requires learning probabilistic behavior, learning to predict uncertain future, learning strategies to outsmart competitors, learning to cooperate, and others. All these learning problems typically involve empirical function induction from observed data.

Optimization in the absence of sufficient prior domain knowledge to guide search directions is often called black-box optimization (BBO). Typical BBO algorithms like evolutionary algorithms (Holland, 1975) and simulated annealing (Kirkpatrick et al., 1983) sample the search space and make inductive decisions in order to explore the promising parts of the search space. The optimization algorithms need to guess intelligently about the landscape using the samples taken from the search domain. This requires inductive detection of function structure.

Learning a function from data is indeed a common problem in many other domains like data mining, software engineering, pattern recognition, and signal processing. Like all these problem domains, the emergence of adaptive living organisms through evolutionary search critically depends on function induction. Natural evolution can be viewed as a search for evolving systems that can adapt and survive the competition of natural selection. If this perspective is correct, then the living organisms must have an efficient mechanism to learn functions from observed data. Indeed, there exist many facts that corroborate this observation. The role of neural networks in our brain is now widely recognized to be an important mechanism for learning functions from observed data (Rumelhart et al., 1986). This paper suggests that there may be an alternate mechanism for inducing function structure in the evolutionary operators of living organisms. The following section argues this possibility.

3 Fitness Evaluation Through Gene Expression

In nature, the evaluation of the “fitness” of an organism is believed to take place through the expression of the DNA through the construction of proteins. This paper suggests that this process can be viewed from the perspective of function induction. It further advances the line of thought by pointing out that the construction of the representation of the genetic fitness may reveal a unique approach for inducing function structures from observed data. In order to fully appreciate this, we need to understand the biology to some extent. The following discussion presents a brief exposure to the natural gene expression process.

DNA is the primary carrier of the evolutionary information that is transmitted from one generation to another. DNA molecules consist of two long, complementary chains held together by base pairs. DNA consists of four kinds of bases joined to a sugar-phosphate backbone. The four bases in DNA are adenine (A), guanine (G), thymine (T), and cytosine (C). Chromosomes are made of DNA double helices. Bases in DNA helices obey the complementary base pairing rule. T and G pair with A and C, respectively. In other words, if the base at a particular position of a helix is T, then the corresponding base in the other helix should be A.
Figure 1: Transcription.

Evaluation of the fitness of a DNA takes place through a process called gene expression. Expression of genetic information coded in DNA involves construction of the mRNA sequence, followed by that of proteins. The main steps are,

- transcription: formation of mRNA (messenger ribonucleic acid) from DNA
- translation: formation of protein from mRNA
- protein folding

In a particular cell, transcription produces the mRNA from a small portion of the DNA. The mRNA defines another level of representation of the genetic information. It consists of four types of bases joined to a ribose-sugar-phosphodiester backbone. The four bases are adenine (A), uracil (U), guanine (G), and cytosine (C). All the bases defining the mRNA are the same as those in DNA sequences, except that T is replaced by U. As shown in Figure 1, the mRNA is produced from the DNA by RNA Polymerase and the regulatory proteins following the complementary base-pairing rules similar to those in DNA. The RNA Polymerase initiates the transcription at a place on the DNA marked by the promoter region (start site). It splits the DNA double helix and continues generating the mRNA using one of the DNA strands as a template. The RNA Polymerase stops when it finds a termination signal sequence (stop site) in the DNA strand. Note that only a small portion of the DNA strand is transcribed and different cells may transcribe different regions of the DNA for producing proteins. Figure 1 presents a schematic diagram of the transcription process.
The mRNA acts as the template for protein synthesis. A protein is defined by a sequence of amino acids, joined by peptide bonds. The mRNA is transported to the cell cytoplasm for producing protein in the ribosome. There exists a unique set of rules that defines the correspondence between nucleotide triplets (known as codons) and the amino acids in proteins. This is known as the genetic code. Each codon is comprised of three adjacent nucleotides in a DNA chain, and it produces a unique amino acid. Amino acid sequence defines a new representation of the information coded in mRNA.

The next level of representation of genetic information is defined by the 3-dimensional structure of folded proteins. Although amino acid sequences fundamentally define proteins, formation of the three dimensional structure of proteins involves a complex process, often called protein folding. This process involves interaction between multiple amino acid subsequences. Figure 2 shows the different steps of the gene expression process.

Since proteins control almost every important activity in a living organism, at an abstract level, gene expression can be viewed as a process that evaluates the merit or fitness of the DNA. Since different proteins are generated at different cells from different portions of the DNA, fitness evaluation in natural gene expression appears to take place in a distributed fashion. In other words, the fitness evaluation seems to be decomposed into different subfunction evaluations. Figure 3 illustrates this distributed, decomposed evaluation of the evolutionary fitness. Such distributed fitness evaluation is possible under either of the two following conditions:

1. the distributed representation of the fitness function was available to evolutionary mechanism a priori;
2. the distributed representation of the function is inductively constructed from the sample DNAs and their respective fitnesses, observed in nature.

In the following sections, we shall adopt the latter possibility and explore it in the context of our basic understanding about function induction. However, first let us review the existing literature in evolutionary computation that considers the computational role of gene expression.
4 Gene Expression in Existing Models of Evolutionary Computation

Although today gene expression is not emphasized very much in most of the popular models of evolutionary computation, several researchers realized its importance. The importance of the computational role of gene expression was realized by Holland. Holland (1975) described the dominance operator as a possible way to model the effect of gene expression in diploid chromosomes. He also noted the importance of the process of protein synthesis from DNA in the computational model of evolution. Despite the fact that, traditionally, dominance maps are explained from the Mendelian perspective, Holland made an interesting leap by connecting it to the synthesis of protein by gene signals, which today is universally recognized as gene expression. He realized the relation between the dominance operator with the “operon” model of the functioning of the chromosome (Jacob and Monod, 1961) in evolution and pointed out the possible computational role of gene signaling in evolution (Holland, 1975).

Several other efforts have been made to model some aspects of gene expression. Diploidy and dominance have also been used elsewhere (Bagley, 1967; Brindle, 1981; Hollstien, 1971; Rosenberg, 1967; Smith, 1988). Most of them took their inspiration from the Mendelian view of genetics. The under-specification and over-specification decoding operator of messy GAs has been viewed as a mechanism similar to gene signaling (Goldberg et al., 1989). The structured genetic algorithm (Dasgupta and McGregor, 1992) also shares motivations from the gene expression; it uses a structured hierarchical representation in which genes are collectively switched on and off. This provides the search algorithm with a richer representation and helps better capture properties of the
landscape. Kauffman (1993) offered an interesting perspective of the natural evolution that realizes the importance for gene expression. However, Kauffman’s work does not explain the process in basic computational terms on analytical grounds and does not relate the issue to the complexity of the search process. The “neutral network” theory (Reidys and Fraser, 1966) considers sequence-to-structure mapping from the perspective of random graph construction. This work investigates the phase transition properties of biomolecules and their implications in the evolutionary optimization in biopolymers. This role of random genotype-to-phenotype mappings in reducing the chance of being trapped in the suboptimal regions of the search space is explored in Shackleton et al. (2000). Experimental approaches to constructing genetic regulatory networks are presented in Morohashi and Kitano (1999).

The role of random genotype-to-phenotype mappings in reducing the chance of being trapped in the suboptimal regions of the search space is explored in Shackleton et al. (2000). Experimental approaches to constructing genetic regulatory networks are presented in Morohashi and Kitano (1999).

There also exists a body of literature that investigates the evolution of the genetic code. An algebraic model of the evolution of the genetic code is presented in Hornos and Hornos (1993). This work searches for symmetries in the genetic code and points out the existence of a unique approximate symmetry group compatible with the codon assignments. The main idea behind this work is to view the evolution of the genetic code as an iterative process of representation decomposition. The genetic code is viewed as a 64-dimensional representation decomposed into several sub-representations with respect to different subgroups. The number of amino acids corresponds to the number of sub-representations, and the number of codons for any amino acid corresponds to the dimension of that sub-representation. An extension of this work using Lie superalgebra is presented in Bashford et al. (1998). Additional work on the different biological theories on the evolution of the genetic code can be found elsewhere (Beland and Allen, 1994; Fukuchi et al., 1994).

An empirical study of genetic programming using artificial genetic code is presented in Keller and Banzhaf (1996). The grammatical evolution-based automated programming approach (Freeman, 1996; Ryan and Collins, 1998; Wong and Leung, 1995) is another interesting effort to incorporate gene expression-based techniques for designing advanced evolutionary algorithms. An empirical investigation of the role of such “non-coding” segments (introns) in genetic search can be found in Wu and Lindsay (1995). A simulation of the expression of the genome in living cells and other related activities is reported in Kennedy (1999).

The following section lays the foundation of the contribution of this paper. It presents a short review of orthonormal basis functions in the function space that will be useful in developing the proposed algorithms.

5 Distributed and Decomposed Representation of Functions

Describing a function first requires choosing a representation. Therefore, learning a function description also requires selecting a representation. In this paper, we will consider representing functions using a weighted summation of certain kinds of functions, called basis functions. Let \( \Xi \) be a set of basis functions. Let us index the basis functions in \( \Xi \) and denote the \( k \)th basis function in \( \Xi \) by \( \Psi_k \). Let \( I \) be the set of all such indices of the basis functions. A function \( f(x) \) can be represented as,

\[
f(x) = \sum_{k \in I} w_k \Psi_k(x),
\]

(1)

Evolutionary Computation Volume 9, Number 1
where $\Psi_k(x)$ denotes the $k$th basis function, and $w_k$ denotes the corresponding coefficient. Although the structure of such functions may appear simple, a properly chosen set of basis functions should be able to represent any given function in the above form.

The objective of function induction can now be viewed as the task to generate a function, $\hat{f}(x) = \sum_{k \in I} \hat{w}_k \Psi_k(x)$, that approximates $f(x)$ from a given data set; $I$ denotes a subset of $I$; $\hat{w}_k$ denotes the approximate estimation of the coefficients $w_k$. For a given basis representation, the underlying inductive task can be viewed as the problem of computing the non-zero, significant (not negligible) coefficients $\hat{w}_k$s. Let us illustrate this using a simple example. Consider a function of Boolean variables, $x_1, x_2, \text{ and } x_3$. We can write,

$$f(x_1, x_2, x_3) = w_0 + w_1 \Psi_1(x_1) + w_2 \Psi_2(x_2) + w_3 \Psi_3(x_3) + w_4 \Psi_4(x_1 x_2) + w_5 \Psi_5(x_1 x_3) + w_6 \Psi_6(x_2 x_3) + w_7 \Psi_7(x_1, x_2, x_3),$$

where, $w_i$s are constant coefficients; $\Psi_i(\cdot)$s are monomial basis functions like Walsh functions (Beauchamp, 1975; Walsh, 1923) to be defined later. At this point, all we need to note is that we can write $f$ as a linear sum of a set of basis functions, weighted by the corresponding basis coefficients. In other words, computation of $f(x_1, x_2, x_3)$ can be performed by computing different subfunctions that require only a subset of the feature set $x_1, x_2, x_3$. Although computation of $\Psi_7(x_1, x_2, x_3)$ requires information about all the three variables, if the value of $w_7$ is close to zero, then we can decompose the computation of $f(x_1, x_2, x_3)$ into different components that require only partial information about the feature set.

Distributed evaluation of a fitness function is possible only when we have its representation using a set of basis functions and their corresponding coefficients. The distributed fitness evaluation in gene expression should be no exception. The efficacy of a certain portion of the DNA depends on the 3-dimensional structure of the protein it produces; since the shape of proteins depends on different physical factors such as energy, bond properties, and others, the set of basis functions used in nature are likely to be functions of these physical factors. However, the exact mathematical structure of these basis functions is not yet known. The algorithms presented in this paper can be formulated using different functionally complete basis sets.

However, in the following discussion, we choose to work with Walsh basis functions because of their existing connections to the field of genetic algorithms (Bethke, 1981; Bridges and Goldberg, 1991; Forrest and Mitchell, 1991; Goldberg, 1989a; Goldberg, 1989b; Goldberg, 1990; Heckendorn and Whitley, 1999; Khuri, 1994; Oei, 1992; Rana et al., 1998). Walsh basis is functionally complete over the space of all Boolean strings. In other words, it can represent any function that can be defined over the space of Boolean strings. The following discussion offers a brief overview of Walsh representation.

Walsh functions (Beauchamp, 1975; Walsh, 1923) are orthogonal functions that found applications in many different fields such as signal processing, image analysis, and others. Like Fourier, Laplace, and other transformations, Walsh functions are often used to represent a problem-solving task in a convenient form. Application of Walsh transformation (WT) in understanding genetic algorithms was first noted by Bethke (1976). Further investigation of this approach can be found elsewhere (Forrest and Mitchell, 1991; Goldberg 1989a, 1989b; Heckendorn and Whitley, 1999; Liepens and
Traditionally, the Walsh functions are used for representing real-valued functions of binary variables. However, they can be easily extended to higher cardinality representation, as shown elsewhere (Oei, 1992). Although the main arguments of the following discussion can be extended for higher cardinality representations, in this paper we shall restrict ourselves to binary variables.

The Walsh basis set is comprised of $2^L$ Walsh functions, where each basis function is defined as follows:

$$\psi_j(x) = (-1)^{\langle x, j \rangle},$$

where $j$ and $x$ are binary strings of length $L$. In other words, $j = j_1, j_2, \cdots, j_L$, $x = x_1, x_2, \cdots, x_L$, and $j, x \in \{0, 1\}^L$. $\psi_j(x)$ can either be 1 or -1. The string $j$ is called a partition. The order of partition $j$ is the number of 1s in $j$. A Walsh function depends on only those $x_i$s only when $j_i = 1$. Therefore, a partition can also be viewed as a representation of a certain subset of $x_i$s; every unique partition corresponds to a unique subset of $x_i$s. If a partition $j$ has exactly $\alpha$ number of 1s, then we say partition is of order $\alpha$, since the corresponding Walsh function is a function of only those variables corresponding to the 1s in the partition $j$. A function $f(x)$ can be written using the Walsh basis functions as follows:

$$f(x) = \sum_j w_j \psi_j(x),$$

where $w_j$ is the Walsh Coefficient (WC) corresponding to the partition $j$. We can easily derive the following expression for WCs exploiting the orthonormality property of Walsh basis functions:

$$w_j = \frac{1}{2^L} \sum_x f(x) \psi_j(x)$$

We note from Equation 3 that a function can be expressed as a linear sum of the Walsh functions, each weighed by the corresponding Walsh coefficients. The Walsh coefficient $w_j$ can be viewed as the relative contribution of the partition $j$ to the function value of $f(x)$. Therefore, the absolute value of $w_j$ can be used as the “significance” of the corresponding partition $j$.

Partitions with non-zero Walsh coefficients capture the underlying structure of the given problem. For example, consider a function $f(x_1, x_2, x_3, x_4) = f_1(x_1, x_2) + f_2(x_3, x_4)$. In the Walsh representation, the values of $w_{1111}$, $w_{1110}$, and any such other Walsh coefficient corresponding to partitions involving a pair of variables, one each from the two linearly decomposable partitions, are zero. Walsh functions can be used for representing any functions of Boolean variables. If a function has dependencies among the features, then its Walsh representation will reflect that, and the function can be evaluated by computing the Walsh functions corresponding to the non-zero Walsh coefficients. If every variable in a function depends on every other variable, then in the general case, the Walsh representation will have all the $2^L$ terms. On the other hand, if at most some $k$ variables interact with each other, all the Walsh coefficients corresponding to partitions with more than $k$ number of 1s will be zero. This class of problems will be called problems with order-$k$ level of interaction. Detection of such interaction among the genes is traditionally called linkage learning in the genetic algorithms literature. A perspective of linkage learning using Walsh representation has been proposed elsewhere (Kargupta and Bandyopadhyay, 2000; Thierens, 1999b).
If the distributed fitness evaluation of the DNA through gene expression is a result of evolutionary search, then evolution must have a mechanism for constructing such a distributed representation of the fitness function. Walsh basis functions offer one way to decompose and distribute the fitness evaluation. Construction of such a representation in a relatively small period of time (in the evolutionary scale) is unlikely to happen unless it is fundamentally possible to do so in rigorous computational ground. The following section identifies a class of algorithms to efficiently construct such a representation.

6 Polynomial-Time Construction of Decomposed Representation

As we saw in the previous section (Equation 4), computation of a single Walsh Coefficient requires information about all the $2^\ell$ domain members. Clearly this cannot be done in time, polynomial in $\ell$. In order to make the problem tractable, we are going to assume that the problem has bounded variable interaction of order-$k$. This is a reasonable assumption for many practical domains. This results in a sparse Walsh representation. In general, function induction is fundamentally difficult if the orthonormal representation is not sparse (Kushilevitz and Mansour, 1993). Even if the problem has bounded order of interaction, explicit computation of WCs using Equation 4 requires exponential time. Another possibility is to generate \( m = \sum_{z=0}^{k} (\ell_z) \) different members from the domain and solve a large \((m \times m)\) linear system of equations in order to compute the \( m \) possibly non-zero terms in the Walsh representation. Although it is theoretically possible in polynomial time, it is not clear how, and if at all, the evolutionary search is equipped with any mechanism to do it this way. In the following discussion, we explore an alternate way to compute individual WCs in an efficient manner.

From Equation 3 we can write,

\[
 f(x)\psi_i(x) = \sum_j w_j \psi_j(x) \psi_i(x) 
\]

Let \( i \) be a partition and \( S(i) \) be the set of all strings that satisfies the following conditions:

1. every member of \( S(i) \) has the same values at all positions where there is a 0 in \( i \) and
2. the substring defined by the positions corresponding to 1s in \( i \) is unique in every member of \( S(i) \).

Let us denote the invariant values at the positions, corresponding to 0s in \( i \) by \( T \) and call it the template.\(^1\) For example, if \( i = 001100 \), then one possible choice of \( S(i) \) may be \{000000, 001000, 000100, 001100\}. \( |S(i)| \) is the size of the set \( S(i) \) and \( |S(i)| = 2^k \) when the partition \( i \) is of order-$k$. In this case, the template \( T = 00\_00 \). Since there exists different such \( S(i) \)s depending on the choice of \( T \), we choose to use the symbol \( S_T(i) \) to denote the set \( S(i) \) with respect to certain template \( T \), wherever needed. Now we can write,

\[
 \sum_{x \in S_T(i)} f(x)\psi_i(x) = \sum_j w_j \sum_{x \in S_T(i)} \psi_j(x) \psi_i(x) 
\]

\[
 = \sum_{j \in J(i)} w_j \sum_{x \in S_T(i)} \psi_j \oplus i (x) 
\]

\[
 = \sum_{x \in S_T(i)} w_j \sum_{j \in J(i)} \psi_j \oplus i (x) = \sum_{x \in S_T(i)} f(x)\psi_i(x), \quad (5)
\]

\(^1\)The idea of using templates in the current context drew motivations from the messy GA literature (Goldberg et al., 1989).
where $J(i)$ denotes the set of all partitions that completely subsumes partition $i$. In other words, every partition in $J(i)$ must have a 1 at every location where there is a 1 in $i$. $j \oplus i$ denotes the partition defined by the Boolean XOR between $j$ and $i$.

The following section presents a simple, deterministic algorithm to compute the Walsh representation in polynomial time for problems with bounded non-linear dependencies among the search variables.

### 6.1 DR1: A Simple Deterministic Algorithm

Since we can choose any invariant set of values for the template, let us consider a special case where all the required values of the template are set to 0. Let us denote the $S_T(i)$ corresponding to this special case template $T = 0$ by $S_0(i)$. Note that the example that we gave earlier follows this case. Equation 5 can be specialized for this case as follows.

$$
\sum_{j \in J(i)} w_j = \frac{1}{|S_0(i)|} \sum_{x \in S_0(i)} f(x) \psi_i(x)
$$

Equation 6 can be used to efficiently detect the significant Walsh coefficients of the function with bounded non-linearity. Recall that for functions with at most $k$ variables non-linearly interacting with each other, all the WCs corresponding to partitions with more than $k$ number of 1s will be zero. Therefore, for all partitions $i$ with more than $k$ number of 1s, the total number of non-zero coefficients in $J(i)$ is zero. For any partition with the order equal or more to $k$, we can write,

$$
w_i = \frac{1}{|S_0(i)|} \sum_{x \in S_0(i)} f(x) \psi_i(x)
$$

Note that the computation of $w_i$ requires only $2^k$ evaluations of $f(x)$. The bounded non-linearity property can therefore be exploited to compute the WCs using only a small number of evaluations ($2^k$) compared to the usual $2^\ell$ evaluations needed using the regular approach. Once we compute all the $k$th order non-zero WCs using Equation 7, lower order coefficients can be computed using the known higher order coefficients and Equation 5. The main algorithmic steps of this technique can be summarized as,

1. select $k \leq \ell$, some constant that bounds the highest order non-linearity of the given problem;
2. compute the order-$k$ WCs using Equation 7;
3. use Equation 5 and the already evaluated order-$k$ WCs to compute order-$(k-1)$ WCs; continue this process iteratively for $(k-1), (k-2), \cdots 1$-order WCs; note that no additional function evaluation is needed after the order-$k$ WCs are computed.

This technique can find all the non-zero WCs of a problem with order-$k$ non-linearity in polynomial time using $2^k \binom{\ell}{k}$ function evaluations. The computation of individual WCs simply requires computing the average of $2^k$ fitness values and some addition and subtraction. Although the complexity of this algorithm is polynomial in $\ell$ for problems with constant $k$, it is fairly expensive for many practical problems. The following section presents a very efficient, non-deterministic algorithm.
6.2 DR2: An Efficient, Non-deterministic Algorithm

This section presents an algorithm to construct the distributed orthonormal representation by evaluating Equation 5 for a set of randomly generated templates. A random choice of template generates a particular assignment of signs of the corresponding Walsh term in Equation 5. For the special case $T = 0$, every $\psi_{\text{B}i}(x) = 1$. If all the coefficients corresponding to partitions that subsume a given partition $i$ are zero, then the left hand side of Equation 5 must produce a zero for any choice of templates. If it returns a non-zero value then at least one of the coefficients in $S_T(i)$ is non-zero. The proposed algorithm makes use of this observation.

The algorithm searches for significant coefficients in the lattice of all partitions. The main steps are as follows:

1. Consider the lattice of all partitions, partially ordered based on the order of the partitions.
2. Start from the partition with zero order, i.e., the one with no fixed bit.
3. Traverse the lattice using a chosen order (e.g., depth-first or breadth-first). Initialize $\alpha = 0$. At each node, do the following $m$ number of times:
   (a) Randomly generate a template.
   (b) Compute $\sum_{x \in S_T(i)} f(x) \psi_i(x)$ at each node; if it returns a non-zero significant number, then exit loop.
   (c) Otherwise, set $\alpha = \alpha + 1$.
4. If $\alpha < m$ continue traversing new nodes using the chosen search strategy.
5. If $\alpha = m$, discard all the nodes subsumed by the current node from the scope of future search for nodes with significant coefficients.

The parameter $m$ is user given. When $m$ is large enough such that every possible template is considered for the test, Equation 5 is guaranteed to detect any non-zero coefficient corresponding to partitions that subsume the partition $i$. The reason behind this is the following. Note that for an $\ell$-bit problem and an order-$k$ partition $i$, there are $2^{\ell-k}$ possible choices of templates. On the other hand, there are $2^{\ell-k}$ different partitions that subsume the partition $i$, and therefore there are that many WCs. Clearly, we have $2^{\ell-k}$ unknown Walsh coefficients and $2^{\ell-k}$ different templates can generate that many equations. If all the tests or all the different templates return zero, then we have a set of $2^{\ell-k}$ homogeneous equations. In that case, the unknown WCs must take only the trivial solution, i.e., all of them are zeros. Although the deterministically correct solution can be obtained only when we enumerate all the templates, a small value of $m$ appears to suffice for many practical cases for the following reason. Let $\nu$ be the proportion of templates at a given node for which $\sum_{x \in S_T(i)} f(x) \psi_i(x) = 0$ despite the existence of non-zero contributing coefficients in the summation. Therefore, the probability that all of the $m$ experiments will return zero, and thereby discard the node, is $\nu^m$. If $\nu < 1$, then the probability of such mistakes goes down exponentially with $m$. So a small value of $m$ should be sufficient for all practical purposes. In fact, randomized sampling of a few templates was sufficient for all the experimental results presented later in this paper. The next section illustrates this algorithm using an example.

Consider the lattice of all sixteen partitions for a four-bit problem as shown in Figure 4. We are interested in a function for which only the partitions, encircled with solid
lines, have non-zero WCs. The proposed algorithm first performs the significance test at all nodes for order-1 partitions. For this case, the test is expected to produce a non-zero value at each of these nodes, indicating that the partition is subsumed by some partitions with non-zero WC. Next the algorithm considers the nodes at the third level from top in Figure 4. Since all the order-2 partitions subsume some set of order-1 partitions, and all the order-1 partitions returned non-zero value for our test, we need to apply our test at every node corresponding to the order-2 partitions. Our test is expected to return non-zero values only for 1100, 1001, and 0101. Next we consider every unique order-3 partition that subsumes at least one of the partitions 1100, 1001, and 0101 but does not subsume any of the partitions 1010, 0110, and 0011. There is only one such partition, and that is 1101. We again apply our test at this node and expect to observe that the test returns a zero value. Next we backtrack to the order-2 partition level and note that the outcomes of the tests we performed earlier at each of these nodes are essentially the WC of the corresponding node, since the coefficient of the partition 1101 is zero. Similarly, we continue to backtrack and compute the coefficients of every non-zero partition. This technique requires $O(2^k r)$ objective function evaluations, where $r$ is the total number of non-zero WCs in its Walsh representation. The algorithm needs to evaluate all the nodes corresponding to the order-2 partitions. Therefore, its complexity should be at least quadratic in $\ell$. For problems where $r$ is $O(\ell^2)$ and $k$ is a constant, the total number of objective function evaluations is $O(\ell^2)$.

Induction of a function in distributed representation is important in many domains like machine learning, optimization, data mining, and others. The algorithms proposed in this section can be applied to each of these application domains. The following section considers its application in optimization.

7 Application of DR2 in Linkage Learning

Evolutionary algorithms are widely used for optimization applications. Scalable optimization in absence of adequate knowledge about the search space requires efficient detection of the regularity and the underlying structure of the search space. The proposed
algorithm can be efficiently used for this purpose. Detection of such structure has been traditionally called linkage learning (Kargupta and Bandyopadhyay, 2000) in the genetic algorithms literature.

Linkage learning algorithms try to detect the structure of the search space using similarity-based equivalence relations or partitions. If a set of search variables together contribute significantly to the objective function, then the quality of the solution will also largely depend on the values set to the these variables. Therefore detection of such a mutually interacting cluster of variables (i.e., the linkage among the variables) is important in optimization. Genetic algorithms-based optimization is no exception. Since the space of all such subsets of variables or partitions is typically exponential in problem size ($\ell$), efficient detection of such significant variable interaction is very important for the scalability of the genetic algorithm.

For hard problems, simple GAs may not provide scalable performance because of this. It has been shown elsewhere (Thierens and Goldberg, 1993; Thierens, 1999a) that the time needed for the convergence to the optimal solution grows exponentially with the problem size in absence of the knowledge regarding the important partitions. The genetic algorithm community has realized the importance of linkage learning in the scalability of the genetic algorithm. A growing number of linkage-learning techniques (Baluja and Davies, 1997; Bandyopadhyay et al., 1998; Deb, 1991; Goldberg et al., 1989, 1993; Harik, 1997; Kargupta, 1996, 1997a; Kargupta and Sarkar, 1999; Mühlenbein and Rodriguez, 1997; Mühlenbein and Mahnig, 1999; Mühlenbein and Paaß, 1996; Thierens, 1999a) are becoming available. The following part of this section presents a brief review of some of the earlier efforts in linkage learning.

The history of linkage learning efforts dates back to Bagley’s (1967) dissertation. Bagley used a flexible representation and the so-called inversion operator for adaptively clustering the related genes. Bagley did not conclude in favor of the use of inversion for linkage learning. Rosenberg (1967) also investigated the possibility of learning linkage by evolving the probability of choosing a location for crossover. Frantz (1972) investigated the utility of the inversion operator and confirmed that inversion is too slow and not very effective. Schaffer and Morishima (1987) introduced a set of flags in the representation. These flags were used for identifying the set of genes to be used for crossover points. They noted the formation of certain favorite crossover points in the population that corroborated their hypothesis regarding the need for detecting gene linkage. Goldberg and Bridges (1990) confirmed that lack of linkage knowledge can lead to failure of GAs for difficult classes of problems, such as deceptive problems. Additional efforts on linkage learning GAs can be found elsewhere (Levenick, 1991; Paredis, 1995).

The messy GA (mGA) (Deb, 1991; Goldberg et al., 1989, 1990) tries to detect linkage by explicitly evaluating schemata (Holland, 1975) using a template (a solution string that is used for filling up the missing values corresponding to the wild card characters of a schema) and selecting good schemata through a selection-only primordial stage. This is a heuristic-based approach, and often its application is computationally expensive. The fast messy GA (Goldberg et al., 1991) offered some reduction in computational cost of linkage learning. As noted earlier, the idea of templates in the algorithms proposed in the current paper draws motivation from the templates in messy GAs.

Several new techniques have recently appeared. The gene expression messy GA (GEMGA) introduced elsewhere (Bandyopadhyay et al., 1998; Kargupta, 1996, 1997a; Kargupta and Bandyopadhyay, 1998) offered a technique for linkage learning. The GEMGA makes use of a local perturbation technique to identify the genes that are critical for high fitness. Once the genes are identified in the local context, they are put to-
together in clusters and tested for global performance. Good clusters define the genetic linkage. This class of algorithms is also based on the local-perturbation heuristics. However the authors reported scalable $O(\ell^2)$ performance.

Harik introduced the linkage learning GA (LLGA) (Harik and Goldberg, 1996; Harik, 1997). The LLGA tries to learn linkage by exploiting the exchange crossover operator and the probabilistic expression based representation. The LLGA appears to work nicely particularly for problems in which construction of a linear ordering among the partitions is not so difficult. In a recent work (Lobo et al., 1998), relations between compressed introns and the LLGA representation is discussed. Smith and Fogarty (1996) reported a technique for evolving genetic linkage and exploiting it for adapting the recombination strategy. They reported superior performance of their linkage learning evolutionary algorithm over the traditional GA.

A linkage learning approach similar to the GEMGA can be found elsewhere (Kemenade, 1996). This approach makes use of a GEMGA-like filtering approach to detect the good partitions and schemata. In a recent work (Munetomo and Goldberg, 1999), a new linkage learning algorithm LINC is proposed. The LINC algorithm works by checking second order non-linearity. It considers feature pairs and performs $O(\ell^2)$ experiments. Let us define $\delta f_i = f(...x_i...) - f(...x_i...)$, $\delta f_j = f(...x_j...) - f(...x_j...)$, and $\delta f_{ij} = f(...x_ix_j...) - f(...x_ix_j...)$, where $x_i$ is a Boolean variable and $x_i = 1 - x_i$. The LINC algorithm detects linkage by exploiting the fact that if two features $i$ and $j$ have pairwise nonlinearity, then $|\delta f_{ij} - \delta f_i - \delta f_j| > \epsilon$, where $\epsilon$ is a constant.

A principal component, analysis-based construction of representation for performing evolutionary optimization has been developed elsewhere (Voigt and Mühlenbein, 1995). Kazadi (1997) proposed a similar approach to explicitly detect generalized schemata that he calls conjugate schemata. His approach proposed a second-order, non-linearity detection technique in the continuous space, similar to what the recently proposed LINC algorithm uses in the discrete Boolean space. A different technique for detecting the significant relations using a dependency tree-based approach has been proposed elsewhere (Baluja and Davies, 1997). This approach also exploits the second-order non-linearity and estimates a second-order approximation of the underlying relations; they reported superior performance of their algorithm over other techniques that do not explicitly try to search and exploit relations.

The approach proposed in Mühlenbein and Paas (1996), Mühlenbein and Rodríguez (1997), and Mühlenbein and Mahnig (1999) is based on estimation of the underlying distribution function. However, since distribution estimation is a hard problem faced in many domains, the approach assumes prior knowledge about the structure of the distribution and reduces the task to estimation of the distribution parameters with known structure. However, their recent effort in this area addressed the issue of learning the search-space structure.

The possibility of linkage learning using Walsh coefficients was first noted elsewhere (Kargupta, 1998; Kargupta and Bandyopadhyay, 2000). Recently, Thierens (1999a) also investigated the possibility of using the KM algorithms (Kushilevitz and Mansour, 1993) for estimating the Walsh coefficients. However, the author reported a scaling problem of the KM approach, since it requires a large number of function evaluations for correctly estimating the coefficients. The DR2 algorithm proposed in this paper shares similarities with the KM approach. Both explore the space of all partitions in a kind of divide-and-conquer approach; both of them are randomized algorithms. However, DR2 imposes a different lattice structure in the space of all partitions that draws motivations from the traditional notion of schema and templates. Unlike the KM-based
approach, DR2-based linkage learning simply tries to detect the existence of any non-zero coefficient down the path rooted at the current node.

The randomized algorithm DR2 can be directly used for constructing the Walsh representation in $O(2^k r)$ time. Once the representation is constructed, we can easily identify the linkage among the genes by noting the relative magnitude of the coefficients. If a partition has a large magnitude, the corresponding variables are strongly linked, and they contribute to the objective function in a significant manner. On the other hand, if a partition has a zero or relatively negligible coefficient, then it does not contribute significantly to the objective function. Therefore, we can first run the proposed algorithm to detect the linkage and then make use of the linkage information to guide the genetic search in a fashion similar to earlier techniques (Bandyopadhyay et al., 1998; Kargupta 1996, 1997a, 1997b, 1998; Kargupta and Bandyopadhyay, 1998; Kargupta et al., 1997). In this paper, we focus only on the former part since that is the primary challenge in front of the linkage-learning genetic algorithms. We have applied the proposed algorithm for detecting the linkage for several test functions. The following section reports the experimental results.

8 Experimental Results

The experimental results presented here report only the cost of detecting the underlying structure since it has already been shown elsewhere (Bandyopadhyay et al., 1998; Kargupta, 1996, 1997a, 1997b, 1998; Kargupta and Bandyopadhyay, 1998; Kargupta et al., 1997) that we can solve such problems using recombination and selection-like operators easily once the structure is correctly detected. These papers reported that the scalability of these techniques critically depends on the complexity of the linkage learning algorithm. The results presented in the following sections demonstrate that the proposed randomized algorithm can be successfully used for linkage learning in any genetic algorithms for solving many optimization problems with bounded dependency in $O(\ell^2)$ time.

The experimental testbed is comprised of problems with bounded dependency. All the objective functions are decomposable to a set of overlapping and non-overlapping subfunctions. For all test problems, at most 5 variables can interact with each other. For all functions with overlapping subfunctions, any given subfunction shares one variable with one other subfunction. No prior knowledge about such decomposability is provided to the linkage-learning algorithm. The proposed algorithm DR2 is used for generating the decomposable representation. The experiments made use of two different ways to pick up the template. In the first case, the template for computing the test at every node of the lattice was selected from prior preliminary experiments. In the second case, a randomly generated template was used. The objectives behind presenting the results for these two different cases are: (1) to demonstrate that reducing a constant number of tests at every node does not change the underlying complexity and (2) if available, prior knowledge about the template can be incorporated into the algorithm for reducing the cost by a constant factor. A template of all 1s and all 0s are chosen for the former case. For the latter case, 10 randomly selected templates are used, which turned out to be sufficient for all test functions. DR2 could detect linkages without any mistake for all test functions using either one of these templates.

TRAP: A Deceptive Function

The deceptive trap (Ackley, 1987) function is defined as:

- $f(x) = k$ if $u = k$;
- $f(x) = k - 1 - u$ otherwise; where $u$ is the unitation variable, or the number of 1s
Table 1: Functions odd(0) and even(0) return true if the number of 0s in $x$ are odd and even, respectively. Odd(1) and even(1) are analogously defined.

<table>
<thead>
<tr>
<th>Mühlenbein</th>
<th>GW2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f(x) = 4$ if $x = 00000$</td>
<td>$f(x) = 10$ if $u=0$</td>
</tr>
<tr>
<td>$= 3$ if $x = 00001$</td>
<td>$= 8$ if $u=k$</td>
</tr>
<tr>
<td>$= 2$ if $x = 00011$</td>
<td>$= 7$ if $u=1$ and odd(0)</td>
</tr>
<tr>
<td>$= 1$ if $x = 00111$</td>
<td>$= 2$ if $u=1$ and even(1)</td>
</tr>
<tr>
<td>$= 0$ if $x = 01111$</td>
<td>$= 4$ if $u=k-1$ and odd(1)</td>
</tr>
<tr>
<td>$= 3.5$ if $x = 11111$</td>
<td>$= 3$ if $u=k-1$ and even(1)</td>
</tr>
<tr>
<td>$= 0$ otherwise.</td>
<td>$= 0$ otherwise.</td>
</tr>
</tbody>
</table>

Table 2: (left) Massively multimodal function and (right) GW1; $u$ denotes the number of 1s in the string. The symbol # denotes the don’t care position.

<table>
<thead>
<tr>
<th>Multi-modal</th>
<th>GW1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f(x) = u+2 \times f'(x)$ where, $f'(x) = 1$ if odd($u$) $= 0$ otherwise</td>
<td>$f(x) = 4$ if $x = 1 # 1 # 0$ $= 8$ if $x = 1 # 0 # 0$ $= 10$ if $x = 0 # 1 # 0$ $= 0$ if $x = 0 # 1 # 0$</td>
</tr>
</tbody>
</table>

in the string $x$, and $k$ is the length of the subfunction. If we carefully observe this trap function, we shall note that it has two peaks. One of them corresponds to the string with all 1s (the global optima) and the other is the string with all 0s (the suboptimal solution). The objective function is constructed by concatenating several deceptive trap functions one after another. The overall objective function is simply a weighted, linear summation of the individual subfunctions. For uniformly scaled problems, all the weights are set to 1. For non-uniformly scaled problems, weights vary through $1, 2, \cdots, \mu$, with the weight for the leftmost subfunction set to 1 and rightmost set to $\mu$. For $\ell = 200$ and $k = 5$, the overall function contains 40 subfunctions; therefore, an order-5 bounded, 200-bit problem has $2^{40}$ local optima, and among them, only one is globally optimal. As the problem length increases, the number of local optima exponentially increases. Figure 5 (top) shows the growth of number of objective function evaluations with respect to increasing number of search variables. Preselected templates are used for these experiments.

MUH: The Mühlenbein Function

This function is defined in Table 1. The global optima is the string of all 0s, while all the strings having a number of trailing 1s constitute the local optima. Unlike the case for TRAP, here the building block corresponding to the global optima has a significant amount of overlap with the local optima. Figure 5 (bottom) shows the growth of number of objective function evaluations with respect to increasing number of search variables. Preselected templates are used for these experiments.

GW1: Goldberg-Wang Function 1

This function is defined in Table 2. Figure 6 (top) shows the growth of number of objective function evaluations with respect to increasing number of search variables.
Figure 5: The number of function evaluations vs. problem size in detecting linkage in case of (top) uniformly scaled non-overlapping TRAP and (bottom) overlapping MUH using selected template. At most five variables can be mutually dependent.

GW2: Goldberg-Wang Function 2
This function is defined in Table 1. Figure 6 (bottom) shows the growth of number of objective function evaluations with respect to increasing number of search variables.

MULTI: A Massively Multi-modal Function
This is a massively multimodal function of unitation where the global optima is a string of all 1s (assuming that length of the subfunction is odd). It is defined in Table 2. This function resembles a one-max function with “bumps.” Figure 7 shows the growth of number of objective function evaluations with respect to increasing number of search variables.
The experimental results presented here clearly demonstrate the scalable $O(\ell^2)$ performance of the proposed randomized algorithm. Since the proposed approach is based on a well-grounded theoretical foundation, it should be equally applicable to other discrete objective functions. The following section discusses the future work and its connections to the biology.
Figure 7: The number of function evaluation vs. problem size in detecting linkage in case of both uniformly scaled non-overlapping and overlapping MULTI with 10 randomized templates. No major difference in the number of function evaluation is noted between the non-overlapping and overlapping cases. At most five variables can be mutually dependent.

9 Future Research Directions

The proposed stochastic algorithm for decomposed representation construction scales up when the number of mutually dependent genes is not greater than some small constant. Although this may be true for many fitness functions, it is easy to construct functions that do not satisfy this condition. Therefore, it is quite natural to ask the validity of this assumption particularly in the context of gene expression. We need to justify this, and we have reasons to believe that the justification may come from the way the gene expression process works. As noted earlier, this process involves several representation transformations. The translation (transformation of mRNA to protein) is one among them. This transformation is controlled by the genetic code. Our recent work (Kargupta, 1999, 2000) extended our approach by exploring the effect of genetic code-like representation transformations in the domain of binary strings. This paper shows that genetic code-like transformations introduce an interesting property in the representation of a genetic fitness function. It points out that such adaptive transformations can convert some functions with an exponentially large description in Walsh basis to one that is highly suitable for polynomial-size approximation. Such transformations can construct a Walsh representation with only a polynomial number of low-order terms that are exponentially more significant than the rest when fitter chromosomes are given more copies through a redundant, equivalent representation. This is a very desirable property (Kushilevitz and Mansour, 1993; Jackson, 1995) for efficient function-induction from data, which is a fundamental problem in learning, data mining, and optimization. This also means that in such representations, higher order interactions among the variables are negligible. We still need to identify the class of functions for which such transformations will work. We also need to develop techniques for constructing such transformations in an evolutionary algorithm.
Representation Construction

One alternate approach to the randomized technique proposed here is to transform the representation of the given objective function in such a way that all the Walsh coefficients are non-negative. In that case, the summation of any subset of such coefficients will be equal to zero only when the coefficients are individually zero. From Equation 4 we can write,

\[ w_j = \sum_{x \in \Omega^+(j)} f(x)\psi_j(x) + \sum_{x \in \Omega^-(j)} f(x)\psi_j(x), \]  

where \( \Omega^+(j) \) and \( \Omega^-(j) \) define the set of all strings for which \( \psi_j(x) = 1 \) and -1, respectively. Clearly there exists a decomposition of any function \( f(x) = f_1(x) - f_2(x) \) in such a way that both \( f_1 \) and \( f_2 \) have non-negative WCs. Note that this requires controlled generation of xs such that the corresponding basis functions for a given \( j \) return either all positive or all negative. This simple construction shows that it is always possible to transform a given function so that all the WCs are non-negative. A simple way to do that is to translate the \( f(0) \) value by some large enough positive constant \( \rho \). Since \( \psi_j(0) = 1 \), such translation will translate every WC by the amount \( \frac{\rho}{2} \). If \( \rho \) is large enough, then all the WCs will be non-negative. Although this makes our algorithm applicable, the problem is that this may destroy the underlying sparseness of the coefficients. In other words, coefficients that were originally zero will become non-zero because of the translation. This may give rise to an exponential value of \( r \), and as a result, the algorithm may no longer remain computable in polynomial time.

An alternate approach to address this problem is to manipulate the representation in such a way that the coefficients become non-negative without destroying the sparseness. It is not clear how to do that. However, it is interesting to observe that complementary alphabet transformations (like what we see in transcription) offer some control over the basis functions that may be useful.

Let \( \beta \) be a binary string, and let \( T \) be a complementary operator that maps 1 to 0 and vice versa. Let \( T_\beta(x) \) be the complementary transformation applied on the \( \beta \)-partition of \( x \). In other words, \( T_\beta(x) \) complements only those positions of \( x \) where there is a 1 in \( \beta \). Let us now consider the difference between the Walsh function \( \psi_\beta(T_\beta(x)) \) and \( \psi_\beta(x) \). Recall that \( \psi_\beta(x) \) can be either -1 or 1. It turns out that \( \psi_\beta(T_\beta(x)) = \psi_\beta(x) \) when the number of transcribed features is even; on the other hand, \( \psi_\beta(T_\beta(x)) = -\psi_\beta(x) \) when the number of transcribed features is odd. This can be understood as follows. Let \( g \) be the number of 1s in \( \beta \); \( g_1 \) and \( g_0 \) be the numbers of 1s and 0s in \( \pi \) within the partition defined by the 1s in \( \beta \). The number of ones in \( T_\beta(x) \) within the partition \( \beta \) is \( g_1' = g_0 = g - g_1 \). Now if both \( g \) and \( g_1 \) are even numbers, then \( g_1' \) must be even. On the other hand, if \( g \) and \( g_1 \) are even and odd numbers, respectively, then \( g_1' \) must be odd. In other words, the sign of \( \psi_\beta(\pi) \) does not change when \( \pi \) is transformed to a different string by applying \( T \) over an even number of bits. Consider \( \psi_{0110}(1011) = -1 \). Now if we apply \( T \) to the underscored bits, we get \( \psi_{0110}(1101) = -1 \). A similar rationale can be developed for the case when \( g \) is odd.

Now consider a transformation of the representation of the domain in such a way that every \( x \) is represented by \( T_1(x) \), where \( 1 \) is an \( \ell \)-bit string of 1s. Such representation flips the sign of the WC corresponding to every odd-order partition. However, it does not destroy the sparseness; WCs with a value of zero, remain zero. We may be able to exploit such complementation-based representation transformation techniques for computing the summation of all non-negative coefficients. However, this is only our hypothesis, since this is similar to the physical representation transformations in transcription at an abstract level.
The current implementation of the proposed algorithm works in a centralized fashion. In other words, it generates the distributed, decomposed representation by searching in a centrally constructed lattice of partitions. Work on extending this algorithm to a hierarchical, distributed computing-based approach is also on progress. In this approach, the partitions defined by different subsets of features are explored in parallel. This work is likely to make the proposed approach suitable for distributed computing environments.

10 Conclusions

This paper notes that fitness evaluation in natural evolution takes place in a distributed fashion in different living cells through the process of gene expression. It argues that such distributed evaluation of the fitness is only possible when the function can be correctly decomposed into different subfunctions. Unless such a representation was available a priori, we have every reason to believe that there must be an efficient mechanism to compute such a distributed, decomposed representation. It develops techniques to do that in polynomial time. It proposes an efficient randomized algorithm to construct such representation of a function when the number of mutually interacting variables is bounded by a constant. Although we could have presented the algorithms without any reference to the gene expression process, we chose not to do so because of our continuing effort to understand this process from the perspective of evolutionary search. We think a presentation from a pure algorithm design perspective without any reference to biology will unnecessarily restrict the scope of our research.

It is important to note that the requirement of bounded order of interaction has deep implication in polynomial complexity search (Kushilevitz and Mansour, 1993); without that, common machine learning and optimization will be quite useless in the general case, as long as polynomial-time computation is desired. For example, consider a decision tree (Quinlan, 1986). Even such a popular inductive learning algorithm may engage in an exponential-time tree construction in absence of this requirement, in the worst case, producing a tree that is no better than table-look-up. Bounded order of dependence among the search variables guarantees a tree of bounded depth and thereby a polynomial-time computability. This assumption is reasonable and most importantly practical. However, as noted earlier, genetic code-like transformations may further extend the scope of the proposed algorithms by constructing a representation of some functions where the higher order coefficients are negligible. For problems that satisfy this condition, the proposed algorithm guarantees linear-time performance with respect to \( r \) with a probabilistic notion of correctness. Although this paper did not directly relate the proposed randomized approach to a specific mechanisms of gene expression, this work and the previous papers along this line have established that polynomial-time computation of the distributed representation of the objective function is possible under reasonable assumption. Now that we have a strong foundation, the next step is to investigate the implications of the specific transformations in gene expression.

Acknowledgments

This work was supported by the United States National Science Foundation Grant IIS-9803660.

References


H. Kargupta and B.-H. Park


